INFORMATION AND COMPUTATION 97, 1–22 (1992)

The Parallel Complexity of Finite-State Automata Problems*

SANG CHO AND DUNG T. HUYNH

Computer Science Program, University of Texas at Dallas, Richardson, Texas 75083

The goal of this paper is to study the exact complexity of several important problems concerning finite-state automata and to classify the degrees of ambiguity of nondeterministic finite-state automata. Our results are as follows: (1) Minimization of deterministic finite automata is NC^1-complete for **NL**. (2) Testing whether the degree of ambiguity of a nondeterministic finite automaton is exponential, or polynomial, or bounded is NC^1-complete for **NL**. (3) Checking whether a given nondeterministic finite automaton is unambiguous or k-ambiguous is NC^1-complete for **NL**, where k is some fixed constant. (4) The bounded nonuniversality problem for nondeterministic finite automata (which is the problem of deciding whether $L(M) \cap \Sigma^{\leqslant n} \neq \Sigma^{\leqslant n}$ for a given nondeterministic finite automaton M and a unary integer n) is log-space complete for **NP**. (5) The bounded nonuniversality problem for unambiguous finite automata is in **DET** (the class of problems NC^1-reducible to computing the determinants of integer matrices), and for deterministic finite automata, it is NC^1-complete for **NL**. (6) The inequivalence problems for unambiguous and k-ambiguous finite automata are both in **DET**, where k is some fixed constant.

0. INTRODUCTION

Following the pioneering papers by Meyer and Stockmeyer (1972), Stockmeyer and Meyer (1973), and Stockmeyer (1974), many works have been done in the study of the complexity of decision and computational problems concerning deterministic and nondeterministic finite automata (DFA and NFA for short). The goal of this line of research was to classify the computational complexity of problems according to the complexity classes **P**, **NP**, **PSPACE**, and the polynomial-time hierarchy. The results obtained have contributed to our understanding of the intrinsic complexity of decision problems in formal language and automata theory. A theorem

* This research was partially supported by the National Science Foundation under Grant DCR-8696097 and by the Organized Research Awards Program of the University of Texas at Dallas.

0890-5401/92 $3.00

stating that some problem belongs to **P**, however, does not tell us whether that problem can be solved by an ultrafast parallel algorithm. The purpose of this paper is, therefore, to classify more precisely the complexity of several important problems concerning DFAs and NFAs in terms of parallel complexity classes (as introduced and studied by S. Cook (1985)). The problems we investigate in this paper are the following: (1) The minimization problem for DFAs, (2) the problem of testing whether the degree of ambiguity of an NFA is exponential, polynomial, or bounded, (3) the problem of checking whether a given NFA is unambiguous or k-ambiguous, where k is some fixed integer (that is not an input parameter), (4) the inequivalence problems for unambiguous finite automata (UFA for short) and k-ambiguous NFAs, where k is fixed.

Minimization of DFAs is certainly an important and well-known problem which has been studied extensively. In fact, as this problem is so often encountered in compiler construction, a lot of effort has been done in designing efficient sequential algorithms. Since there exists an $O(n \log n)$ algorithm for minimizing DFAs (Hopcroft, 1971), it is interesting to show that this problem is in **NC**. In fact, we prove that it is even NC^1-complete for **NL** and hence is in $\mathbf{NC}^2$. (**NL** is the class of problems solvable by nondeterministic log-space bounded Turing machines, and **NC** the class of problems solvable in polylog time by parallel algorithms using only a polynomial number of processors.)

The study of the degree of ambiguity of NFAs has received much attention in the last several years. Stearns and Hunt (1985) gave polynomial-time algorithms for the problems of testing whether a given NFA is unambiguous and whether an NFA is k-ambiguous, where k is a fixed integer not in the input. Subsequently, Ibarra and Ravikumar (1986) provided an exponential time algorithm for checking whether the degree of ambiguity of an NFA is exponential. The most recent result is by Weber and Seidl (1986), who showed that testing whether the degree of ambiguity of an NFA is bounded is in **P**. From the works of Ibarra and Ravikumar (1986), Reutenauer (1977), and Weber and Seidl (1986), it follows that the degree of ambiguity of an NFA is either exponential, or polynomial, or bounded. Here we show that the problems of testing whether the degree of ambiguity of an NFA is exponential, or polynomial, or bounded, or k-bounded are all NC^1-complete for **NL**, where k is a fixed constant.

Regarding the inequivalence problem for NFAs, Stockmeyer and Meyer (1973) provided a **PSPACE**-completeness result. The nonuniversality problem for NFAs (which is the problem of deciding for a given NFA M whether $L(M) \neq \Sigma^*$) is also log-space complete for **PSPACE**. Here, we are concerned with the bounded version of the nonuniversality problem for NFAs which is defined as follows: Given an NFA M and a unary integer n, decide whether $L(M) \cap \Sigma^{\leq n} \neq \Sigma^{\leq n}$, where $\Sigma^{\leq n}$ denotes the set of strings

of length $\leqslant n$ over Σ. (Let BNU denote the bounded nonuniversality problem.) It turns out that the complexity of BNU is significantly lower than that of the inequivalence problem. As a matter of fact, we show that BNU for NFAs is log-space complete for **NP**, BNU for UFAs is in **DET**, and BNU for DFAs is NC^1-complete for **NL**. (**DET** is the class of problems NC^1-reducible to computing the determinants of integer matrices; Cook, 1985.) We also consider the problem of computing the lexicographically first string which witnesses the inequality $L(M) \cap \Sigma^{\leqslant n} \neq \Sigma^{\leqslant n}$ for a given NFA M and a unary integer n (LFWITNESS for short). We show that LFWITNESS for NFAs is in Δ_2^P $(=\mathbf{P}^{\mathrm{NP}})$ and that it is both **NP**-hard and **CoNP**-hard. Further, we also show that LFWITNESS for UFAs is in **P**, and LFWITNESS for DFAs is NC^1-complete for **NL**. The complexity of the inequivalence problem reduces considerably if we restrict the class of NFAs under consideration. In fact, Stearns and Hunt (1985) show that the inequivalence problems for UFAs and k-ambiguous NFAs are both in **P**, where k is some fixed integer not in the input. We show that these problems both belong to **DET** (and hence are in $\mathbf{NC}^2$).

This paper is organized as follows. Section 1 contains definitions and notations used in the paper. In Section 2, we classify the complexity of the minimization problem for DFAs. In Section 3, we give a complete characterization of the degrees of ambiguity for NFAs and show that all decision problems concerning the degrees of ambiguity of NFAsd are NC^1-complete for **NL**. Section 4 contains complexity results for the bounded nonuniversality problem (BNU) and the problem of computing the lexicographically first witness string (LFWITNESS) for NFAs, UFAs, and DFAs. In this section, we also show that the inequivalence problem for UFAs is in **DET**. Finally, Section 5 contains some concluding remarks. We hope that the results in this paper provide a more precise classification of the complexity of decision and computational problems for finite automata. In view of recent developments in the theory of parallel computation, we believe that such classification is interesting.

1. PRELIMINARIES

For the sake of completeness we introduce in this section basic notions and concepts that are used in this paper. We assume familiarity with standard notions and concepts in automata-based complexity theory. **P**, **NP**, **PSPACE** have the usual meanings. We refer the reader to Hopcroft and Ullman (1979) for further details, and Garey and Johnson (1979) for a definition of the polynomial time hierarchy. In the following, we first review some important definitions from the theory of parallel computation (cf. Cook, 1985, for a detailed discussion).

DEFINITION 1.1. A problem R (with size parameters r and s) is a family $\langle R_n \rangle$ of binary relations such that $R_n \subseteq \{0,1\}^{r(n)} \times \{0,1\}^{s(n)}$. A circuit family $\langle \alpha_n \rangle$ is said to solve the problem R if and only if the function $\langle f_n \rangle$ computed by $\langle \alpha_n \rangle$ realizes R in the following sense: For each n and each x in $\{0,1\}^{r(n)}$, if $R_n(x, y)$ holds for some y, then $R_n(x, f_n(x))$ holds.

DEFINITION 1.2. We say that a circuit family $\langle \alpha_n \rangle$ is log-space uniform if there is a deterministic $\log n$-space bounded Turing machine that computes a description of the circuit α_n, where n is given as a unary integer. $\mathbf{NC}^k$ is the class of all problems R solvable by a log-space uniform circuit family $\langle \alpha_n \rangle$ with size $(\alpha_n) = n^{O(1)}$ and $\text{depth}(\alpha_n) = O((\log n)^k)$. $\mathbf{NC} = \bigcup_k \mathbf{NC}^k$.

DEFINITION 1.3. A problem R is $\mathbf{NC}^1$-reducible to S (written $R \leqslant_{NC^1} S$) if and only if there is a log-space uniform family $\langle \alpha_n \rangle$ of circuits for solving R, where $\text{depth}(\alpha_n) = O(\log n)$, and α_n is allowed to have oracle nodes for S. An oracle node for S is a node with some sequence $\langle y_1, ..., y_r \rangle$ of input edges and a sequence $\langle z_1, ..., z_s \rangle$ of output edges whose values satisfy $S(y_1 \cdots y_r, z_1 \cdots z_s)$. In defining depth of α_n, such an oracle node counts as depth $\lceil \log(r+s) \rceil$.

A problem R is $\mathbf{NC}^1$-hard for the class $\mathbf{C}$ if and only if $S \leqslant_{NC^1} R$ for all S in $\mathbf{C}$. Further, R is said to be $\mathbf{NC}^1$-complete for $\mathbf{C}$ if and only if R is $\mathbf{NC}^1$-hard for $\mathbf{C}$ and $R \in \mathbf{C}$.

DEFINITION 1.4. Let *INTDET* denote the problem of computing $\det(A)$ for a given $n \times n$ matrix A of n-bit integer entries. The class **DET** is defined as

$$\mathbf{DET} = \{R \mid R \leqslant_{NC^1} \mathit{INTDET}\}.$$

As noted in Cook (1985), $\leqslant_{NC^1}$ is reflexive and transitive, and $\mathbf{NC}^k$ is closed under $\leqslant_{NC^1}$ for all $k \geqslant 1$. Further, the following inclusions show the relations between parallel complexity classes:

$$\mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{DET} \subseteq \mathbf{NC}^2 \subseteq \mathbf{NC} \subseteq \mathbf{P}.$$

Let **NL** be the class of functions computed by nondeterministic log-space bounded Turing machines. As usual we can view a decision problem as a zero–one function. Thus, we sometimes regard **NL** as the class of languages accepted by nondeterministic log-space bounded Turing machines. We use the following result.

PROPOSITION 1.5 (Immerman, 1988; Szelepcsényi, 1988). **NL** *is closed under complement.*

In the following we introduce some basic definitions concerning finite automata.

DEFINITION 1.6. A nondeterministic finite automaton (NFA) is a 5-tuple $M=(Q, \Sigma, \delta, p_0, F)$, where

(1) Q is a finite set of states,

(2) Σ is a finite set of input symbols,

(3) δ is a function from the set $Q \times \Sigma$ into the set of subsets of Q,

(4) $p_0 \in Q$ is the initial state,

(5) $F \subseteq Q$ is the set of final states.

A path π of length m for x from p to q in M is a string $\pi = q_0 x_1 q_1 x_2 q_2 \cdots x_m q_m \in Q(\Sigma Q)^*$ so that $x = x_1 x_2 \cdots x_m \in \Sigma^*$, $p = q_0 \in Q$, $q = q_m \in Q$, $x_i \in \Sigma$, and $q_i \in \delta(q_{i-1}, x_i)$ for $1 \leqslant i \leqslant m$. A path π from p to q is an accepting path if p is the initial state and q is a final state.

An unambiguous finite automaton (UFA) is a special NFA in which there is at most one accepting path for each string $x \in \Sigma^*$. A deterministic finite automaton (DFA) is a special NFA in which δ is a function from $Q \times \Sigma$ into Q.

DEFINITION 1.7. Let $M=(Q, \Sigma, \delta, q_0, F)$ be an NFA and $p \in Q$. A state p is called a *useful* state if there is an accepting path which includes p; otherwise p is said to be *useless*. If no state of M is useless, then M is called *reduced*.

Let $M=(Q, \Sigma, \delta, q_0, F)$ be a DFA and $p, q \in Q$. We say p is *equivalent* to q $(p \equiv q)$ if and only if for each string x, $\delta(p, x) \in F$ if and only if $\delta(q, x) \in F$. p is *inequivalent* to q if there exists an $x \in \Sigma^*$ such that $\delta(p, x)$ is in F and $\delta(q, x)$ is not, or vice versa. A state p is an *accessible* state if there is a path from q_0 to p, otherwise p is said to be *inaccessible*.

2. THE COMPLEXITY OF THE MINIMIZATION PROBLEM FOR DFAS

Minimization of DFAs is the problem of computing for a given DFA an equivalent one that has a minimum number of states. In Hopcroft (1971) one can find an $O(n \log n)$ time algorithm for this problem. We show in this section that minimizing the number of states of DFAs is NC^1-complete for **NL**. Without loss of generality we only consider DFAs with the input alphabet $\Sigma = \{0, 1\}$.

THEOREM 2.1. *The minimization problem for DFAs is NC^1-complete for* **NL**.

The proof of Theorem 2.1 is carried out in Lemmas 2.2–2.4 below. Lemma 2.2 shows that the minimization problem is NC^1-hard for **NL**, whereas Lemmas 2.3 and 2.4 together prove that this problem is in **NL**. Before showing the upper bound, we note that the problem of testing for a given DFA M whether a state q is accessible is already NC^1-complete for **NL**. Therefore, we consider only DFAs whose states are all accessible, and show that, even under this restriction, the problem is still NC^1-hard for **NL**.

LEMMA 2.2. *The minimization problem for DFAs without inaccessible states is NC^1-hard for* **NL**.

Proof. As usual, we reduce an **NL**-complete problem to the minimization problem. A familiar problem that is NC^1-complete for **NL** is the graph accessibility problem, denoted by GAP. GAP is the problem of deciding for a given (directed) graph G and two vertices s and g whether there is a path from s to g in G. A special case of GAP is the accessibility problem for graphs whose vertices have outdegrees $\leqslant 2$, denoted by 2GAP. It can easily be seen that 2GAP is also NC^1-complete for **NL**. We reduce 2GAP to the minimization problem for DFAs without inaccessible states as follows.

Let (G, s, g) be an instance of 2GAP where $G=(V, E)$ and s is the start vertex and g is the goal vertex. Without loss of generality let $V = \{1, 2, ..., n\}$, $s = 1$, $g = n$. First, we construct a DFA $M_1 = (Q_1, \Sigma, \delta_1, s, \{g\})$ as follows:

$$Q_1 = V, \Sigma = \{0, 1\}, \text{ and } \delta_1 \text{ is defined by: for each vertex } i,$$

(1) if outdegree$(i)=2$: let j, k $(j<k)$ be two vertices adjacent to vertex i

$$\delta_1(i, 0) = j \qquad \text{and} \qquad \delta_1(i, 1) = k,$$

(2) if outdegree$(i)=1$: let j be the vertex adjacent to vertex i

$$\delta_1(i, b) = j \qquad \text{for all} \quad b \in \Sigma,$$

(3) if outdegree$(i)=0$:

$$\delta_1(i, b) = i \qquad \text{for all} \quad b \in \Sigma.$$

The DFA M_1 is well defined and it is clear that there is a path from s to g in G if and only if $L(M_1)$ is not empty.

Next we construct a DFA $M=(Q, \Sigma, \delta, q_1, \{g\})$ from M_1 so that every state is accessible. The construction of M_1 is as follows:

$Q = Q_1 \cup Q_2$, where $Q_2 = \{q_1, q_2, ..., q_n\}$, and δ is defined by:

$$\begin{aligned}
&\delta(p, b) = \delta_1(p, b) && \text{for } p \in Q_1, b \in \Sigma,\\
&\delta(q_i, 0) = i && \text{for } 1 \leqslant i \leqslant n,\\
&\delta(q_i, 1) = q_{i+1} && \text{for } 1 \leqslant i \leqslant n-1,\\
&\delta(q_n, 1) = q_n.
\end{aligned}$$

It can easily be seen that every state in Q is accessible, and that if $i < j$ then q_i and q_j are inequivalent, since there is a string $x = 1^{n-j}0$ such that $\delta(q_i, x)$ is not in F but $\delta(q_j, x)$ is in F. We now show the following

CLAIM. *There is no path from s to g in G if and only if the minimum state DFA $M' = (Q', \Sigma, \delta', q_1', F')$ for M has a state $p \in Q'$ so that $\delta'(q_1', 0) = p$, $\delta'(p, 0) = p$, $\delta'(p, 1) = p$, and p is not in F'.*

Proof of Claim. Suppose there is no path from s to g in G and let S be the set of states reachable from state s. Then all states in S are equivalent. Therefore, in the construction of a minimum state DFA M' for M, S is "collapsed" into a single state p in M'. Note that the state q_1 is inequivalent to the state s since there is a path from q_1 to g. Thus the states q_1' and p are inequivalent. Since the state p is not a final state and there is no transition from p to any other state the condition is satisfied. The other direction is obvious. ∎

From the above claim we can easily see that there is a path from s to g in G if and only if there is no $p \in Q'$ so that $\delta'(q_1', 0) = p$, $\delta'(p, 0) = p$, $\delta'(p, 1) = p$, and p is not in F'. This gives us a reduction from 2GAP to the minimization problem. As this recuction is clearly an NC^1 reduction, Lemma 2.2 follows. ∎

The next two lemmas show that the minimization problem is in **NL**.

LEMMA 2.3. *Deciding the inequivalence of two states in a DFA is in* **NL**.

Proof. Lemma 2.3 follows immediately from Proposition 1.5. ∎

To finish the proof of Theorem 2.1, we prove the following lemma.

LEMMA 2.4. *Constructing for a given DFA a minimum state DFA is in* **NL**.

Proof. Let $M = (\{1, 2, ..., n\}, \Sigma, \delta, 1, F)$ be a given DFA. We can construct a minimum state DFA $M' = (\{1, 2, ..., m\}, \Sigma, \delta', 1, F')$ for M using following algorithm:

smallest (i) {* returns the smallest state j equivalent to i in M^*}
  if inaccessible (i) **then** return (0);
  for $j := 1$ **to** i **do**
    if equivalent (i, j) **then** return (j);

newstate (i) {* returns the new state in M' for a given state in M^*}
  $s := 0$;
  for $j := 1$ **to** smallest (i) **do**
    if smallest $(j) = j$ **then** $s := s + 1$;
  return (s);

$m :=$ max {newstate $(i) \mid 1 \leqslant i \leqslant n$};
$F' := \{j \mid j \neq 0$ and $j =$ newstate (i) for some $i \in F\}$;
δ' is defined by:
  δ' (newstate (i), b) = newstate (j) for all $b \in \Sigma$,
  where $\delta(i, b) = j$, newstate $(i) \neq 0$, and newstate $(j) \neq 0$;

Observe that there is a one–one correspondence between the states of M' and the sets of equivalent states of M which are reachable. Hence, the correctness of the algorithm follows. Clearly, this algorithm can be implemented on a nondeterministic log-space bounded Turing machine. This completes the proof of Lemma 2.4. ▮

3. Complexity of Determining the Degree of Ambiguity for NFAs

This section consists of two subsections. In Section 3.1 we introduce two conditions that completely characterize the degrees of ambiguity of NFAs. Using these two conditions, we then show that the degree of ambiguity of an NFA is exponential, or polynomial, or bounded by a fixed integer. Section 3.2 deals with complexity classification for the problem of testing whether the degree of ambiguity of an NFA is exponential, or polynomial, or bounded. We show that this problem is NC^1-complete for **NL**. We also show that determining whether a given NFA is unambiguous or k-ambiguous for a fixed constant k is NC^1-complete for **NL**.

3.1. *Characterization of the Degrees of Ambiguity of NFAs*

The characterization of the degrees of ambiguity of NFAs is based on the previous works by Ibarra and Ravikumar (1986), Reutenauer (1977), and Weber and Seidl (1986). Before stating the two conditions that characterize the degrees of ambiguity of NFAs, we introduce some technical definitions. In the following let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA.

DEFINITION 3.1. We say that a state p is connected with a state q if there are paths from p to q, and from q to p in M. We write p **CON** q if p, q are connected. Clearly, **CON** is an equivalence relation on Q. Let $Q_1, Q_2, ..., Q_k$ be the equivalence classes with respect to **CON**.

OBSERVATION 3.2. If we consider each Q_i, $1 \leqslant i \leqslant k$, as a (hyper) vertex, then the transition diagram of M becomes a directed acyclic graph.

DEFINITION 3.3. The degree of ambiguity of a string x in the NFA M, denoted by $\mathrm{Amb}(M, x)$, is defined as

$$\mathrm{Amb}(M, x) = \text{the number of distinct accepting paths for } x.$$

The degree of ambiguity of the NFA M on strings of length m, denoted by $\mathrm{Amb}(M, m)$, is defined as

$$\mathrm{Amb}(M, m) = \max\{\mathrm{Amb}(M, x) \mid x \in \Sigma^m\}.$$

(Note that the function $\mathrm{Amb}(M, m)$ is well defined.)

DEFINITION 3.4. The degree of ambiguity of the NFA M is said to be

(1) *exponential* if there is a constant $c > 0$ so that

$$\mathrm{Amb}(M, m) \geqslant 2^{cm} \qquad \text{infinitely often;}$$

(2) *polynomial* if there are constants $c_1, c_2 > 0$ and an integer i so that

$$\mathrm{Amb}(M, m) \leqslant c_1 m^i \qquad \text{for all } m, \text{ and}$$

$$\mathrm{Amb}(M, n) \geqslant c_2 m \qquad \text{infinitely often;}$$

(3) *bounded* if there is a constant $c > 0$ so that

$$\mathrm{Amb}(M, m) \leqslant c \qquad \text{for all } m.$$

DEFINITION 3.5. When the ambiguity of M is bounded by some fixed integer k, then we say the NFA M is *k-ambiguous*. If the degree of ambiguity of M is bounded by 1 then it is called an *unambiguous* finite automaton (UFA for short).

We are now in the position to introduce the two conditions that characterize the degrees of ambiguity of NFAs. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA.

CONDITION 1 (Ibarra and Ravikumar, 1986; Reutenauer, 1977). There exist a state $q \in Q$ and strings $u, v, w \in \Sigma^*$ for which there are two distinct paths from q to q labeled by v and $q \in \delta(q_0, u)$, $\delta(q, w) \cap F \neq \varnothing$.

CONDITION 2 (Weber and Seidl, 1986). There are two distinct states $p, q \in Q$ and strings $u, v, w \in \Sigma^*$ so that $p, q \in \delta(p, v)$ and $q \in \delta(q, v)$, $p \in \delta(q_0, u)$, $\delta(q, w) \cap F \neq \varnothing$.

It follows from Ibarra and Ravikumar (1986), Reutenauer (1977), and Weber and Seidl (1986) that the degree of an NFA M is

(1) exponential if M satisfies Condition (1),

(2) polynomial if M satisfies Condition (2), but not Condition (1), and

(3) bounded if M does not satisfy Condition (2).

Therefore we obtain

THEOREM 3.6. *The degree of ambiguity of an NFA is* (1) *exponential, or* (2) *polynomial, or* (3) *bounded.*

The proof of Theorem 3.6 is done through the following four lemmas.

LEMMA 3.7. *If an NFA M satisfies Condition* (1), *then the degree of ambiguity of M is exponential.*

Proof. This follows immediately from Condition (1). ∎

Although the technique in Ibarra and Ravikumar (1986) can be used to prove the following lemma, we provide here a simple proof. Without loss of generality we may assume that M is reduced.

LEMMA 3.8. *If an NFA M does not satisfy Condition* (1), *then there exists a constant $c > 0$ and an integer k so that* $\mathrm{Amb}(M, m) \leqslant cm^k$ *for all m.*

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be the given NFA and Q_i, $1 \leqslant i \leqslant k$, be the equivalence classes on Q with respect to the relation **CON**. Since M does not satisfy Condition (1), for any string $v \in \Sigma^*$ there is at most one path for v from p to q if $p, q \in Q_i$, $1 \leqslant i \leqslant k$. In other words, once the start state p and the end state q are fixed the path for v in Q_i is uniquely determined. Let us consider an accepting path π for an input x of length m:

$$\pi = \pi_1 y_{1,2} \pi_2 y_{2,3} \cdots \pi_{j-1} y_{j-1,j} \pi_j,$$

where π_i is a subpath of π that is contained in Q_{p_i} for some p_i, $1 \leqslant p_i \leqslant k$, and $y_{i-1,i} \in \Sigma$. For $i = 1, \ldots, j$ let

$$\pi_i = q_0^i x_1^i q_1^i x_2^i q_2^i \cdots x_{n_i}^i q_{n_i}^i$$

with

$$x = x_1^1 x_2^1 \cdots x_{n_1}^1 y_{1,2} x_1^2 \cdots x_{n_2}^2 y_{2,3} \cdots y_{j-1,j} x_1^j \cdots x_{n_j}^j,$$

where $q_0^1 = q_0$, $q_{n_j}^j \in F$, $q_h^i \in Q_{p_i}$, and x_h^i, $y_{i-1,i} \in \Sigma$ for $1 \leqslant i \leqslant j$, $1 \leqslant h \leqslant n_i$.

Observe that each of the $y_{i-1,i}$, $2 \leqslant i \leqslant j$, corresponds to a transition from a state in an equivalence class by **CON** to a state in another equivalence class by **CON**. According to Observation 3.2, if we regard the equivalence classes by **CON** as (hyper-) vertices, then the resulting digraph is acyclic. Therefore, j is bounded by k, the number of equivalence classes by **CON**. Thus, we can upper-bound the number of possible accepting paths for x as follows. For each $i = 2, ..., j$, consider the position of $y_{i-1,i}$ in x. Since $|x| = m$, the number of possible combinations of positions of $y_{1,2}, ..., y_{j-1,j}$ in x is bounded by m^k. For a fixed combination of positions of $y_{1,2}, ..., y_{j-1,j}$ in x, each substring $x_1^i \cdots x_{n_i}^i$, $1 \leqslant i \leqslant j$, corresponds to a labeled subpath π_i that is contained in an equivalence class by **CON**. Since M does not satisfy Condition (1), there are no more than $|Q|^2$ possibilities for such a subpath π_i. Remember there are at most k such subpaths. Thus, the total number of possible accepting paths labeled x is bounded by $|Q|^{2k} m^k$, which is bounded by cm^k for some constant c, since $|Q|$ and k are independent of input x. This completes the proof of Lemma 3.8. ∎

LEMMA 3.9. *If an NFA M satisfies Condition* (2), *then there is a constant c so that* $\mathrm{Amb}(M, m) \geqslant cm$ *infinitely often.*

Proof. This follows immediately from Condition (2). ∎

LEMMA 3.10. *If the degree of ambiguity of an NFA M is not bounded, then M satisfies Condition* (2).

Proof. See Weber and Seidl (1986). ∎

3.2. *Complexity of Determining the Degree of Ambiguity for NFAs*

In this subsection, we show that determining the degree of ambiguity for NFAs is NC^1-complete for **NL**.

THEOREM 3.11. *Determining the degree of ambiguity for NFAs is NC^1-complete for* **NL**.

The following two lemmas establish Theorem 3.11. Note that testing whether a state q of an NFA M is useful or not is already NC^1-complete for **NL**. Therefore, we prove Theorem 3.11 under the assumption that the NFA M in the input is reduced.

LEMMA 3.12. *Testing whether a given reduced NFA satisfies Condition* (1) *is NC^1-complete for* **NL**.

Proof. First, one easily sees that a nondeterministic log-space bounded Turing machine can test whether a given reduced NFA satisfies Condition (1).

Second, we want to reduce 2GAP, which is known to be NC^1-complete for **NL** as noted earlier, to the above problem. Let (G, s, g) be an instance of 2GAP where $G=(V, E)$, s is the start vertex, and g is the goal vertex. Let $V=\{1, ..., n\}$. If each edge $e=(i, j)\in E$ satisfies $i<j$, then we call this restricted version of 2GAP *monotone* 2GAP. In the following we briefly describe the reduction from 2GAP to monotone 2GAP. For a given G we make n copies of V and denote them by $V_1, V_2, ..., V_n$, respectively. Now for each edge $(i, j)\in E$ where $i\neq g$ we add an edge from vertex i in V_{k-1} to vertex j in V_k for all $k=2, ..., n$. We then renumber all vertices in the new graph in a natural manner and the resulting graph is an instance of monotone 2GAP. It is not hard to see that there is a path from s to g in G if and only if there is a path of length $n-1$ from s in V_1 to g in V_n. Since NC^1 reducibility is transitive, we can use monotone 2GAP instead of 2GAP to show that certain problem is NC^1-hard for **NL**.

Let (G, s, g) be an instance of monotone 2GAP where $G=(V, E)$, $V=\{1, ..., n\}$, $s=1$, and $g=n$. First, we construct an NFA $M_1=(Q_1, \Sigma, \delta_1, 1, \{n\})$ as follows:

$Q_1=V$, $\Sigma=\{0, 1\}$, and δ_1 is defined by:

(1) outdegree $(i)=2$: let j, k $(j<k)$ be two vertices adjacent to vertex i

$$\delta_1(i, 0)=j \qquad \text{and} \qquad \delta_1(i, 1)=k,$$

(2) outdegree $(i)=1$: let j be the vertex adjacent to vertex i

$$\delta_1(i, b)=j \qquad \text{for all} \quad b\in\Sigma.$$

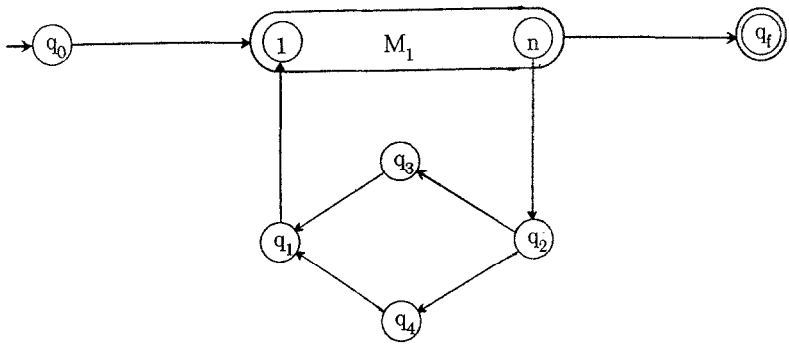
Now, construct a reduced NFA $M=(Q, \Sigma, \delta, q_0, \{q_f\})$ as follows:

$Q=Q_1\cup\{q_0, q_1, q_2, q_3, q_4, q_f\}$, and δ is defined by:

For all $b\in\Sigma$:

$$\delta(q_0, b)=Q_1,$$
$$\delta(q_1, b)=\{1\},$$
$$\delta(q, b)=\delta_1(q, b)\cup\{q_f\} \qquad \text{when} \quad q\in Q_1-\{n\},$$
$$\delta(n, b)=\{q_2, q_f\},$$
$$\delta(q_2, b)=\{q_3, q_4\},$$
$$\delta(q_3, b)=\{q_1\},$$
$$\delta(q_4, b)=\{q_1\}.$$

This construction is depicted in Scheme 1.



SCHEME 1

For this NFA M we can easily verify that the following conditions are equivalent:

(i) there is a path from vertex 1 to vertex n in G,

(ii) M satisfies Condition (1),

(iii) M satisfies Condition (2),

(iv) $L(M)$ is infinite.

Thus, we obtain a reduction from monotone 2GAP to testing whether a reduced NFA satisfies Condition (1). Further, this reduction is an NC^1 reduction. This completes the proof of Lemma 3.12. ▮

LEMMA 3.13. *Testing whether a reduced NFA satisfies Condition* (2) *but not Condition* (1) *is* NC^1*-complete for* **NL**.

Proof. First, one easily observes that testing whether a given reduced NFA satisfies Condition (2) but not Condition (1) is in **NL**.

Second, we want to reduce monotone 2GAP to testing whether a reduced NFA M satisfies Condition (2) but not Condition (1). Note that this **NL**-hardness does not follows from the construction given in Lemma 3.12 above. Let (G, s, g) be an instance of monotone 2GAP, where $G=(V, E)$, $V=\{1, ..., n\}$, $s=1$, and $g=n$. As in the proof of Lemma 3.12, we can build the NFA $M_1=(Q_1, \Sigma, \delta_1, 1, \{n\})$. Now we construct a reduced NFA $M=(Q, \Sigma, \delta, q_0, q_f)$ as follows:

$$Q=Q_1 \cup \{q_0, q_f\}, \text{ and } \delta \text{ is defined by:}$$

For all $b \in \Sigma$:

$$\delta(q_0, b) = Q_1,$$
$$\delta(q, b) = \delta_1(q, b) \cup \{q_f\}, \quad \text{if} \quad q \in Q_1 - \{1, n\},$$
$$\delta(1, b) = \delta_1(1, b) \cup \{1, q_f\},$$
$$\delta(n, b) = \delta_1(n, b) \cup \{n, q_f\}.$$

This construction is depicted in Scheme 2.

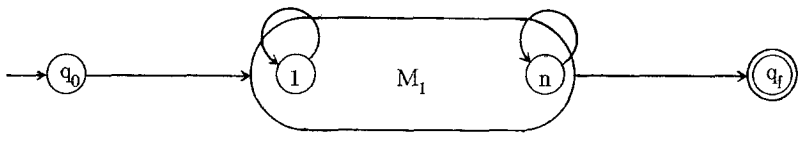
Observe that there is no cycle in the NFA M_1. Thus, it is clear that the reduced NFA M satisfies Condition (2) but not Condition (1) if and only if there is a path from s to g in G. This reduction is obviously an NC^1 reduction. Thus, the proof of Lemma 3.13 is complete. ▮

Using a similar argument we can show that testing whether an NFA is k-ambiguous for some fixed k is also NC^1-complete for **NL**.

LEMMA 3.14. *Testing whether an NFA is k-ambiguous for some fixed integer k is NC^1-complete for* **NL**.

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a given NFA. The following algorithm gives a positive answer when ambiguity of M is greater than or equal to k:

$s_1 := q_0; s_2 := q_0; \ldots; s_k := q_0;$
let mark = array $[1 \cdots k, 1 \cdots k]$ of boolean;
initialize all the entries of mark false;
repeat
  guess a symbol $b \in \Sigma$;
  let r_1 be one of $\delta(s_1, b)$;
  $s_1 := r_1;$
  let r_2 be one of $\delta(s_2, b)$;
  $s_2 := r_2;$
  ...
  let r_k be one of $\delta(s_k, b)$;
  $s_k := r_k;$
  for all the pairs of $i < j$ **do**
    if $s_i \neq s_j$ **then** mark $[i, j]$:= true;



SCHEME 2

```
until mark [i, j] = true for all the pairs i < j;
if any of s_i is not in F then begin
    repeat
        guess a symbol b ∈ Σ;
        let r_1 be one of δ(s_1, b);
        s_1 := r_1;
        let r_2 be one of δ(s_2, b);
        s_2 := r_2;
        ...
        let r_k be one of δ(s_k, b);
        s_k := r_k;
    until s_i ∈ F for all i, 1 ≤ i ≤ k;
end;
write ('yes');
```

Clearly, the above algorithm can be implemented on a nondeterministic log-space bounded Turing machine. Since **NL** is closed under complement, testing whether an NFA M is $(k-1)$-ambiguous is in **NL**. As k can be any fixed integer, we conclude that testing whether an NFA is k-ambiguous is in **NL**.

To show **NL**-hardness, we can easily reduce monotone 2GAP to the complement of testing whether an NFA is k-ambiguous as follows. From an input instance (G, s, g) of monotone 2GAP, we construct the DFA M_1 as in the proof of Lemma 3.12, make $(k+1)$ distinct copies of it, and then connect them together using a new initial and a new final state to form a reduced NFA M: The new initial state is connected to each original initial state of the $(k+1)$ DFAs, and each original final state of the $(k+1)$ DFAs is connected to the new final state. Clearly, M is $(k+1)$-ambiguous, and it is not k-ambiguous if and only if there is a path from s to g in G. ∎

Applying an argument similar to the one in the proof of Lemma 3.14, we obtain

COROLLARY 3.15. *Testing whether an NFA is unambiguous is* NC^1*-complete for* **NL**.

4. COMPLEXITY OF INEQUIVALENCE PROBLEMS FOR UFAS

Complexity classification of equivalence problems for various language classes has received much attention. It is well known that the inequivalence problem for NFAs is log-space complete for **PSPACE** (Stockmeyer and Meyer, 1973) and the inequivalence problem for DFAs is log-space com-

plete for **NL** (Jones and Lien, 1976). As shown by Stearns and Hunt (1985), the inequivalence problem for UFAs is in **P**. In this section, we prove, as the main result, that the inequivalence problem for UFAs is in **DET**. We also consider several interesting problems related to the inequivalence problem, namely the bounded nonuniversality problem and the problem of computing the lexically first witness string. (Recall that the lexical order is defined by: strings are ordered according to length and strings of same length are ordered lexicographically.) These problems are defined as follows.

DEFINITION 4.1. The bounded nonuniversality (BNU for short) for a class of automata **M** is defined as follows:

Instance. An automaton $M \in \mathbf{M}$ with terminal alphabet Σ and a nonnegative unary integer n.

Question. Is $L(M) \cap \Sigma^{\leq n} \neq \Sigma^{\leq n}$?

DEFINITION 4.2. The problem of computing the lexically first witness string for the bounded nonuniversality (LFWITNESS for short) for a class of automata **M** is defined as follows:

Input. An automaton $M \in \mathbf{M}$ and a nonnegative unary integer n.

Output. The lexically first string w of length less than or equal to n which does not belong to $L(M)$ if any; otherwise it is defined to be ∞.

PROPOSITION 4.3. *BNU and LFWITNESS for DFAs are both* NC^1*-complete for* **NL**.

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a given DFA. The following algorithm gives a positive answer if there is a string x of length less than or equal to n which is not accepted by M:

```
count := n;
s := q0;
repeat
    guess a symbol b ∈ Σ;
    s := δ(s, b);
    count := count − 1;
until (s not in F) or (count < 0);
if count ⩾ 0 then write ('yes');
```

Clearly, the above algorithm can be implemented on a nondeterministic log-space bounded Turing machine. Thus, BNU for DFAs is in **NL**.

The following algorithm computes the lexically first witness string x of length less than or equal to n which is not accepted by M:

```
i := −1;
repeat
    i := i + 1;
until BNU(M, i) = yes or i > n;
if i > n then {Write ('∞'); Halt}
else if count = 0 then Halt
else count := i;
s := q0;
repeat
    p := δ(s, 0);
    q := δ(s, 1);
    if there is a path of length ⩽count − 1 from p
          to a state not in F then {s := p; write ('0')}
    else if there is a path of length ⩽count − 1 from q
          to a state not F then {s := q; write ('1')}
    else {write ('error'); Halt}
    count := count − 1;
until p is not in F or q is not in F;
```

Again, this algorithm can be implemented on a nondeterministic log-space bounded Turing machine. Thus, LFWITNESS for DFAs is in **NL**.

Now, to show the lower bound we reduce 2GAP to the BNU problem for DFAs. Let (G, s, g) be an instance of 2GAP where $G = (V, E)$ and $V = \{1, 2, ..., n\}$. We construct a DFA $M = (Q, \Sigma, \delta, s, F)$ as follows:

$$Q = V, \Sigma = \{0, 1\}, F = Q - \{g\}, \text{ and } \delta \text{ is defined by:}$$

(1) outdegree $(i) = 2$: let j, k $(j < k)$ be two vertices adjacent to vertex i

$$\delta(i, 0) = j \qquad \text{and} \qquad \delta(i, 1) = k,$$

(2) outdegree $(i) = 1$: let j be the vertex adjacent to vertex i

$$\delta(i, b) = j \qquad \text{for all} \quad b \in \Sigma,$$

(3) outdegree $(i) = 0$:

$$\delta(i, b) = i \qquad \text{for all} \quad b \in \Sigma.$$

From the construction above we can see that $L(M) = \Sigma^*$ if and only if there is no path from s to g. Thus $L(M) \cap \Sigma^{\leqslant n} \neq \Sigma^{\leqslant n}$ if and only if there is a path from s to g. Since this reduction is clearly an NC^1 reduction, BNU for DFAs is NC^1-hard for **NL**. Finally, to show that LFWITNESS for DFAs is also NC^1-hard for **NL**, we observe that BNU is NC^1-reducible

to LFWITNESS. Hence, LFWITNESS for DFAs is NC^1-hard for **NL**. This completes the proof of Proposition 4.3. ▮

For NFAs the complexity of BNU and LFWITNESS is significantly higher.

PROPOSITION 4.4. *BNU for NFAs is log-space complete for* **NP**.

Proof. To show that BNU for NFAs is **NP**-hard, we simply modify the proof of the **PSPACE**-hardness of the (unbounded) nonuniversality for NFAs in (Stockmeyer and Meyer, 1973), where regular expressions are used to describe the invalid computations of polynomial-space bounded Turing machines. Observe that for a nondeterministic polynomial-time bounded Turing machine M_1, the invalid computations of M_1 on an input string can be described by strings of polynomial length, whereas such strings might have exponential length if they are to describe polynomial-space bounded computations. Now assume that the machine M_1 on inputs of length m operates in exactly $p(m)$ steps, where p is some fixed polynomial. We can compute the exact length of the description of a computation of M_1 on input x of length m, and then convert the regular expression that describes the invalid computations of M_1 on input x to an NFA M. With n being the length of the description of a computation of M_1 on input x of length m, let (M, n) be an input instance of BNU. Then, (M, n) belongs to BNU iff the Turing machine M_1 accepts x. Since this reduction can be carried out by a deterministic log-space bounded Turing machine, we conclude that BNU for NFAs is log-space hard for **NP**. As it is obvious that BNU for NFAs is in **NP**, the proof of Proposition 4.4 is complete. ▮

PROPOSITION 4.5. *LFWITNESS for NFAs is in* Δ_2^P $(=\mathbf{P}^{NP})$. *Further, LFWITNESS for NFAs is both* **NP**-*hard and* **CoNP**-*hard.*

Proof. The following algorithm solves the LFWITNESS problem for NFAs. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a given NFA:

```
i := −1;
repeat
    i := i + 1;
until BNU(M, i) = yes or i > n;
if i > n then {write ('∞'); Halt}
else if count = 0 then Halt
    else count := i;
S := {q0};
```

repeat
  $P := \delta(S, 0)$;
  $R := \delta(S, 1)$;
  build an NFA $M_0 = (Q \cup \{p_0\}, \Sigma, \delta', p_0, F)$ where
  $\delta'(p_0, \varepsilon) = P$ and $\delta'(q, b) = \delta(q, b)$ if $q \in Q$, $b \in \Sigma$;
  if BNU(M_0, count $-$ 1) = yes
    then {$S := P$; write ('0')}
  else begin
    build an NFA $M_1 = (Q \cup \{r_0\}, \Sigma, \delta'', r_0, F)$ where
    $\delta''(r_0, \varepsilon) = R$ and $\delta''(q, b) = \delta(q, b)$ if $q \in Q$, $b \in \Sigma$;
    if BNU(M_1, count $-$ 1) = yes
      then {$S := R$; write ('1')}
    else {write ('error'); halt}
  end;
  count := count $-$ 1;
until count = 0;

Clearly, the above algorithm can be implemented on a deterministic polynomial-time bounded oracle machine with oracle BNU. **NP**-hardness and **CoNP**-hardness of LFWITNESS for NFAs are immediate. This completes the proof of Proposition 4.5. ∎

PROPOSITION 4.6. *BNU for UFAs is in* **DET**.

Proof. Let $M = (Q, \Sigma, \delta, 1, F)$ be a given UFA where $Q = \{1, ..., m\}$. Observe that the number of accepting paths is the same as that of accepted strings. We can compute the number of accepting paths of length n by computing D^n, where $D = (d_{ij})$ is an $m \times m$ matrix whose entries are given by:

$$d_{ij} = \text{the cardinality of } \{b \mid j \in \delta(i, b)\}.$$

Let $E = (e_{ij}) = D^n$. The number of accepting paths of length n is $\sum_{j \in F} e_{1j}$. Clearly, (M, n) belongs to BNU iff there is an integer $k \leqslant n$ so that the number of accepting paths of length $k \neq 2^k$. We can compute D^k for $k = 1, ..., n$ and check whether there is any k so that the number of accepting paths of length $k \neq 2^k$. Since integer matrix powering is in **DET** and the BNU problem for UFAs is NC^1-recudible to integer matrix powering, we conclude that BNU for UFAs is in **DET**. ∎

PROPOSITION 4.7. *LFWITNESS for UFAs is in* **P**.

Proof. We use the same algorithm as that of Proposition 4.5 with the exception that the **NP** oracle is replaced by a **DET** oracle. ∎

We now turn our attention to the main result of this section. Before showing that the inequivalence problem for UFAs is in **DET**, we mention a simple fact about difference equations.

DEFINITION 4.8. Let A be a function from $\mathbf{N}$ to $\mathbf{R}$. We say that A satisfies a homogeneous linear difference equation with constant coefficients of degree n if and only if there exist constants $c_i \in \mathbf{R}$, for $1 \leqslant i \leqslant n$, with $c_n \neq 0$ such that $\sum_{i=0}^{n} c_i A(k+i) = 0$.

The following proposition is a well-known fact.

PROPOSITION 4.9. *Let $A(k)$, $B(k)$ be two homogeneous linear difference equations with constant coefficients of degrees n_1 and n_2, respectively. If $A(k) = B(k)$ for $0 \leqslant k \leqslant n_1 + n_2 - 1$ then $A(k) = B(k)$ for all $k \geqslant 0$.*

THEOREM 4.10. *The inequivalence problem for UFAs is in* **DET**.

Proof. Let $\text{AccPath}_M(k)$ denote the number of accepting paths of length k of a UFA M. Stearns and Hunt (1985) pointed out that $\text{AccPath}_M(k)$ satisfies a homogeneous linear difference equation with constant coefficients (difference equation for short) of degree n, where n is the number of states of M.

Now let M_1, M_2 be two UFAs. From the given UFAq M_1, M_2 we can build a UFA M_3 so that $L(M_3) = L(M_1) \cap L(M_2)$. Clearly, this construction can be done by an NC^1-circuit. We note that if M_1, M_2 are unambiguous then M_3 is unambiguous, too. Therefore, deciding the inequivalence of M_1, M_2 reduces to deciding the strictness of the containments $L(M_3) \subseteq L(M_1)$ and $L(M_3) \subseteq L(M_2)$. Now, $L(M_3) = L(M_1)$ if and only if $\text{AccPath}_{M_3}(k) = \text{AccPath}_{M_1}(k)$ for $0 \leqslant k \leqslant n_3 + n_1 - 1$, where n_3, n_1 are the number of states of M_3, M_1, respectively. (The same observation holds for $L(M_3)$ and $L(M_2)$.)

In Proposition 4.6, we have seen that computing $\text{AccPath}_M(k)$ for a given UFA M and a unary positive integer k is in **DET**. Thus, we can compute AccPath_{M_3}, AccPath_{M_1}, AccPath_{M_2} and easily verify whether they are equal or not for all $k = 0, \ldots, (n_3 + n_2 + n_1 - 1)$, and these computations can be carried out by an NC^1-circuit with a **DET** oracle. We conclude that the inequivalence problem for UFAs is in **DET**. ∎

COROLLARY 4.11. *The inequivalence problem for k-ambiguous NFAs is in* **DET**.

Proof. We can apply the proof technique of Theorem 4.10 together with an argument in Stearns and Hunt (1985). (The details are omitted.) ∎

Remarks. In this section, we have seen that the complexity of the inequivalence and bounded nonuniversality problems and the problem of computing the lexically first witness string depend heavily on the degree of ambiguity of the automata under consideration. For NFAs the inequivalence is **PSPACE**-complete, whereas it is in **DET** for UFAs. The bounded nonuniversality problem for NFAs is **NP**-complete. For UFAs it is in **DET**. Regarding the problem of computing the lexically first witness string we obtain a Δ_2^P upper bound, and **NP**- and **CoNP**-hardness lower bounds. For UFAs the problem is in **P**. Whether the LFWITNESS problem for UFAs is in **NC** or not remains an open question. (We note that it is suggested in the paper "Are Search and Decision Problems Computationally Equivalent?" by Karp, Upal, and Wigderson (1985) that the search problem is probably harder than the decision problem in parallel computation.) Finally, observe that if we are interested in computing any witness string (instead of the lexically first one) for NFAs, then this problem is equivalent to the bounded nonuniversality problem and is therefore **NP**-complete. For UFAs, this problem does not appear to be easier than the LFWITNESS problem.

5. Conclusions

In this paper, we have obtained several results concerning the parallel complexity of some important decision and computational problems for finite-state automata. The minimization problem for DFAs is NC^1-complete for **NL**. Regarding the ambiguity degree of NFAs, we provided two simple conditions which can be used to classify the degree of ambiguity of an NFA. It turns out that the ambiguity degree of an NFA is exponential, or polynomial, or bounded. This result is interesting in its own right. Indeed, it implies, for example, that the ambiguity degree of an NFA cannot be of the form $\theta(n^{\log n})$. We also showed that determining whether the degree of ambiguity of an NFA is exponential, or polynomial, or bounded is NC^1-complete for **NL**. This should not be confused with a result by Chan and Ibarra (1983), who showed that deciding whether an NFA is d-ambiguous with d as an input parameter is **PSPACE**-complete. For the inequivalence problem for UFAs we obtain a **DET** upper bound. **NL**-hardness follows from the **NL**-completeness of the inequivalence problem for DFAs. It would be interesting to close this gap. We note that a refree has pointed out that Kuich (1988), using a different technique, has independently shown that the inequivalence problem for UFAs and k-ambiguous NFAs is reducible to integer matrix powering which is known to be in **DET**. Finally, we think it is worthwhile to show that the problem of computing the lexicographically first witness string for UFAs is in **NC** or that it is **P**-complete.

ACKNOWLEDGMENTS

The authors thank two anonymous referees for helpful remarks that greatly improve the presentation of this paper. We especially thank one of them for pointing out two references (Kuich, 1988; Reutenauer, 1977).

RECEIVED January 19, 1990; FINAL MANUSCRIPT RECEIVED June 27, 1990

REFERENCES

COOK, S. (1985), A taxonomy of problems with fast parallel algorithms, *Inform. and Control* **64**, 2–22.

CHAN, T., AND IBARRA O. (1983), Note on the finite-valuedness problem for sequential machines, *Theoret. Comput. Sci.* **23**, 95–101.

GAREY, M. R., AND JOHNSON, D. S. (1979), "Computers and Intractability, A Guide to the Theory of *NP*-completeness," Freeman, New York.

HOPCROFT, J. E. (1971), An $n \log n$ algorithm for minimizing the states in a finite automaton, *in* "The Theory of Machines and Computations" (Z. Kohavi, Ed.), pp. 189–196, Academic Press, New York.

HOPCROFT, J., AND ULLMAN, J. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, MA.

IBARRA, O. H., AND RAVIKUMAR, B. (1986), On sparseness, ambiguity and other decision problems for acceptors and transducers, *STACS*, 171–179.

IMMERMAN, N. (1988), Nondeterministic space is closed under complement, *SIAM J. Comput.* **17**, 935–938.

JONES, N. D., AND LIEN, Y. E. (1976), New problems complete for nondeterministic log space, *Math. Systems Theory* **10**, 1–17.

KARP, R. M., UPFAL, E. U., AND WIGDERSON, A. (1985), Are search and decision problems computationally equivalent? *in* Proceedings 17th Annual Symp. on Theory of Computing," pp. 464–474.

KUICH, W. (1988), "Finite Automata and Ambiguity," Technical Report 253, Institut für Informationsverarbeitung, Technische Universität Gratz, Austria.

MEYER, A. R., AND STOCKMEYER, L. J. (1972), The equivalence problem for regular expressions with squaring requires exponential space, *in* Proceedings IEEE 13th Annual Symp. on Switching and Automata Theory," pp. 125–129.

REUTENAUER, CHR. (1977), "Propriétés arithmétiques et topologiques de series rationalles en variables non commutatives," Thesis, Université de Paris VI.

STEARNS, R. E., AND HUNT, H. B., III (1985), On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata, *SIAM J. Comput.* **14**, 598–611.

STOCKMEYER, L. J. (1974), "The Complexity of Decision Problems in Automata Theory and Logic," Report TR-133, MIT Project MAC, Cambridge, MA.

STOCKMEYER, L. J., AND MEYER, A. R. (1973), Word problems requiring exponential time, *in* Proceedings Fifth Annual ACM Symp. on the Theory of Computing," pp. 1–9.

SZELEPCSÉNYI, R. (1988), The method of forced enumeration for nondeterministic automata, *Acta Inform.* **29**, 279–284.

WEBER, A., AND SEIDL, H. (1986), On the degree of ambiguity of finite automata, *in* "Lecture Notes in Computer Science," Vol. 233, pp. 620–629.